UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte LUKE MATTHEW BROWNING, KENNETH BERNARD ROZENDAL and SURESH ESWARE WARRIER

Appeal No. 2005-2527
Application No. 09/620,714[1]

ON BRIEF

Before THOMAS, HAIRSTON and SAADAT, Administrative Patent Judges.
SAADAT, Administrative Patent Judge.

DECISION ON APPEAL

This is a decision on appeal from the Examiner's final rejection of claims 1-29, which are all of the claims pending in this application.

We reverse.

BACKGROUND

Appellants' invention is directed to a method and apparatus for debugging programs from a predetermined starting point. According to Appellants, the entire process state of the process

[1] Application for patent filed July 20, 2000.

or processes can be saved at any time in the course of a debugging session such that the execution of the program may be terminated at any time and resumed from the point where the state had been saved (specification, page 3). An understanding of the invention can be derived from a reading of exemplary independent claim 1, which is reproduced as follows:

1. A method in a data processing system for debugging a process from a starting point, comprising:

initiating debugging of the process;

saving a process state in response to a first event to form a stored process state;

retrieving the stored process state in response to a predefined event; and

reinitiating debugging from the stored process state.

The Examiner relies on the following references:

Lenkov et al. (Lenkov)	5,560,009	Sep. 24, 1996
Kato	6,240,529	May 29, 2001 (filed Jun. 3, 1998)
Leask et al. (Leask)	6,412,106	Jun. 25, 2002 (filed Jun. 16, 1999)

Claims 1-6, 11-18 and 23-29 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Kato.

Claims 7-9 and 19-21 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Kato and Lenkov.

Claims 10 and 22 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Kato and Leask.

Rather than reiterate the opposing arguments, reference is made to the briefs and answer for the respective positions of Appellants and the Examiner. Only those arguments actually made by Appellants have been considered in this decision. Arguments which Appellants could have made but chose not to make in the briefs have not been considered (37 CFR § 41.37(c)(1)(vii)).

OPINION

With respect to the 35 U.S.C. § 102 rejection of claims 1-6, 11-18 and 23-29, Appellants argue that the state restoration disclosed in Kato is responsive to a state restoration command currently being issued which is not the same as the claimed "in response to a predetermined event" (brief, page 6). Appellants further point out that Kato actually assists the user in manually selecting from many state storage files the one to be used upon issuing a state restore command (brief, page 8). Appellants also argue that although Kato stores state information when a certain event occurs, restoration of the process requires manual user intervention (id.).

In response, the Examiner relies on state restoration command of Kato (col. 8, lines 29-35) and characterizes issuing

such state restoration command as "a predefined event" which causes the state restoration unit to read in the file and restore the debugged state (answer, page 4). The Examiner further takes the position that the claims do not require automatic intervention and, therefore, a manual user intervention is not precluded (id.).

Appellants disagree with equating the state restoration command as the claimed predefined event while stating that the issuance of such command itself may be an event (reply brief, page 2). Appellants further argue that, however, such command issuance is user-initiated and, at best, is a user-initiated event rather than a "predefined event" (id.)

A rejection for anticipation requires that the four corners of a single prior art document describe every element of the claimed invention, either expressly or inherently, such that a person of ordinary skill in the art could practice the invention without undue experimentation. See Atlas Powder Co. v. Ireco Inc., 190 F.3d 1342, 1347, 51 USPQ2d 1943, 1947 (Fed. Cir. 1999); In re Paulsen, 30 F.3d 1475, 1478-79, 31 USPQ2d 1671, 1673 (Fed. Cir. 1994).

Hoffman relates to debugging a program and restoring a stored system state for re-execution of the program from the

point the storing occurred (col. 1, lines 9-18). The storing of the debugging state occurs upon detecting an event set in advance (col. 5, lines 14-23) while the restoration of the debugged state follows issuing a state restoration command. In particular, the portions of the reference relied on by the Examiner (col. 8, lines 29-35) disclose that upon issuing a state restoration command, storage situation management unit 121 displays a list of currently existing state storage files to be selected (Fig. 4; col. 8, lines 29-32). Although Kato describes file restoration when the state restoration command is issued, the fact that a file to be restored is required to be selected from the list (col. 8, lines 33-35) indicates that the process restoration requires a state restoration command and selection of a state from a file list.

Claim 1 requires retrieving the stored process state in response to a predefined event which is also described as an event set by the programmer in advance as a condition for instructing the debugger to restore the process (specification, pages 15 & 16). What a reference teaches is a question of fact. In re Baird, 16 F.3d 380, 382, 29 USPQ2d 1550, 1552 (Fed. Cir. 1994) (citing In re Beattie, 974 F.2d 1309, 1311, 24 USPQ2d 1040, 1041 (Fed. Cir. 1992)). Here, the Examiner has not pointed to

any condition or predefined event that triggers the step of retrieving the stored process step. Therefore, we find ourselves in agreement with Appellants' assertion that the state restoration command itself is not a predefined event, as required by claim 1 (reply brief, page 2).

In view of the discussion above, we find ourselves in agreement with Appellants that the claimed feature of "in response to a predefined event" is not taught or suggested by Kato. Claims 13 and 25 include similar limitations related to retrieving the saved process state in response to a predefined event which, as discussed above with respect to claim 1, is absent in Kato.

Regarding the remaining independent claims, Appellants argue that Kato does not teach a process having control over at least one child process, as recited in claims 6, 18 and 28 (brief, page 9; reply brief, page 2). Additionally, Appellants point to the feature in claim 29 that requires storing two different process states from two different processes and argue that Kato merely traces a single state file during the debugging process (brief, page 15; reply brief, page 4).

In response, the Examiner argues that the debugging process calls other processes which are child processes, as shown in

Figure 5A, and stores their state in the list (answer, age 6). However, we disagree with the characterization of adding "a function to manage a situation upon storage of a debugged state" (col. 8, lines 16-19) as the claimed "child process" (answer, page 6) since the portions relied on by the Examiner do not show that the process being debugged has control over at least one child process. In fact, as pointed out by Appellants, Kato discloses storing the process state of only a single process.

Similarly, the Examiner relies on saving a debugged state when an event is detected (col. 10, lines 28-41; col. 8, lines 29-32) to conclude that multiple process states are saved which may correspond to both traced and untraced processes, as recited in claim 29 (answer, page 13). Again we remain unconvinced that these portions of Kato teach the claimed storing a process state of a traced process and a process state of an untraced process that are later retrieved to reinitiate debugging the process. Therefore, we agree with Appellants (brief, page 15; reply brief, page 4) that Kato stores the process state of the one process being debugged upon occurrence of a certain event instead of storing the states of two different processes.

Accordingly, since Kato does not teach all the claimed limitations discussed above, the Examiner's burden of providing a

prima facie case of anticipation is not met. Therefore, the 35 U.S.C. § 102 rejection of claims 1-6, 11-18 and 23-29 over Kato cannot be sustained.

Turning now to the 35 U.S.C. § 103 rejection of the remaining claims, we note that the Examiner further relies on Lenkov for teaching data type descriptors (answer, page 21) and on Leask for disclosing that register values are modified before suspending the program (answer, page 22). However, similar to Kato, the Examiner has not pointed to any teachings or suggestions in Lenkov and Leask that relate to retrieving a stored process state in response to a predefined event in order to overcome the deficiencies of Kato as discussed above with respect to claims 1 and 13. Accordingly, we do not sustain the 35 U.S.C. § 103 rejection of claims 7-9 and 19-21 over Kato and Lenkov and of claims 10 and 22 over Kato and Leask.

CONCLUSION

In view of the foregoing, the decision of the Examiner rejecting claims 1-6, 11-18 and 23-29 under 35 U.S.C. § 102 and rejecting claims 7-10 and 19-22 under 35 U.S.C. § 103 is reversed.

REVERSED

JAMES D. THOMAS
Administrative Patent Judge)

KENNETH W. HAIRSTON
Administrative Patent Judge) BOARD OF PATENT APPEALS AND INTERFERENCES

MAHSHID D. SAADAT
Administrative Patent Judge)

MDS/ki

Appeal No. 2005-2527
Application No. 09/620,714

IBM Corp. (YA)
C/O Yee & Associates, PC
P.O. Box 802333
Dallas, TX 75380